Vehicle routing with arrival time diversification

Hoogeboom, Maaike; Dullaert, Wout

published in
European Journal of Operational Research
2019

DOI (link to publisher)
10.1016/j.ejor.2018.11.020

document version
Publisher's PDF, also known as Version of record

European Journal of Operational Research 275 (2019) 93–107

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Production, Manufacturing, Transportation and Logistics

Vehicle routing with arrival time diversification

Maaike Hoogeboom*, Wout Dullaert

Department of Information, Logistics and Innovation, Vrije Universiteit Amsterdam, HV Amsterdam 1081, The Netherlands

ARTICLE INFO

Article history:
Received 8 November 2017
Accepted 8 November 2018
Available online 13 November 2018

Keywords:
Routing
Heuristics
Multiple time windows
Security constraints

ABSTRACT

Unpredictable routes may be generated by varying the arrival time at each customer over successive visits. Inspired by a real-life case in cash distribution, this study presents an efficient solution approach for the vehicle routing problem with arrival time diversification by formulating it as a vehicle routing problem with multiple time windows in a rolling horizon framework. Because waiting times are not allowed, a novel algorithm is developed to efficiently determine whether routes or local search operations are time window feasible. To allow infeasible solutions during the heuristic search, four different penalty methods are proposed. The proposed algorithm and penalty methods are evaluated in a simple iterated granular tabu search that obtains new best-known solutions for all benchmark instances from the literature, decreasing average distance by 29% and reducing computation time by 93%. A case study is conducted to illustrate the practical relevance of the proposed model and to examine the trade-off between arrival time diversification and transportation cost.

1. Introduction

Cash-in-transit (CIT) companies transfer valuable goods to banks, ATMs, and stores. For security reasons and owing to legal regulations, these companies must use varying routes. Interviews with safety managers of CIT companies in the Netherlands indicate that the moment of arrival at the customer and the periods when the vehicle is stationary are the most vulnerable points on the transportation journey. Inspired by this real-life case, this study focuses on alternating the arrival times at each customer and minimizing transportation costs. Additionally, waiting time is not allowed, to minimize the time that an armored truck is stationary.

To the best of the authors' knowledge, there are only three studies focused on alternating arrival times over successive visits to a customer. In Calvo and Cordone (2003), the overnight security service problem was introduced, in which a set of different travelling salesman problem (TSP) solutions should be obtained. This problem was solved by generating different instances for the TSP with time windows: the initial customer time windows were split into four sub-windows, and they were assigned to four subsets of customers in 4! ways. In Yan, Wang, and Wu (2012), a mathematical model was formulated in which the current arrival time at a customer should vary by at least ϵ minutes compared to previous arrival times; moreover, a small case study was considered using CPLEX. In Michallet, Prins, Amodeo, Yalaoui, and Vitry (2014), the vehicle routing problem (VRP) with arrival time diversification was formulated by varying the arrival time at each customer by at least ϵ minutes over P successive visits. This problem was solved in a periodic setting by proposing a heuristic in which waiting time is not allowed. A time-dependent penalty function was used to penalize arrival times that violate the arrival time diversification constraint. The periodic setting and the time-dependent penalty function complicate this method, as a local search operation in one period can also modify the penalties in other periods. Thus, the evaluation of the local search moves is computationally intensive.

In this study, a more efficient and powerful solution method is presented for the same arrival time diversification problem as in Michallet et al. (2014). In the proposed method, a rolling horizon setting of one day is used, whereas in Michallet et al. (2014), the problem was solved in a periodic setting. As CIT orders are received on short notice, a CIT plans its routes on a daily basis, and thus the rolling horizon approach matches this real-life setting. Instead of penalizing previous arrival time intervals as in Michallet et al. (2014), the proposed method deletes the P previous arrival times from the solution space, along with the surrounding bandwidth ϵ. The result is a routing problem with multiple time windows in which each customer is still available for service. Therefore, the problem is modeled as a vehicle routing problem with multiple time windows (VRPMTW) in which the multiple time windows are constructed in a rolling horizon setting by deleting the arrival intervals of previous days.

As waiting time is not allowed, it is not efficient to use existing algorithms for the VRPMTW that allow waiting time

* Corresponding author.
E-mail addresses: m.hoogeboom@vu.nl (M. Hoogeboom), wout.dullaert@vu.nl (W. Dullaert).

https://doi.org/10.1016/j.ejor.2018.11.020
0377-2217/

(e.g., Belhaiza, Hansen, & Laporte, 2014; Hoogeboom, Dullaert, Lai, & Vigo, 2018). Therefore, a novel algorithm is presented for determining whether a departure time from the depot exists, so that no waiting time will occur during a given route. Moreover, this algorithm can also be used to efficiently determine whether local search moves will result in a time window feasible route. A simple iterated granular tabu search is proposed to illustrate the performance of the VRPMTW formulation and the proposed algorithm. Because infeasible solutions are allowed during the search, four separate measures are considered to penalize time window violations, and these measures are compared based on the computational time and solution quality of the metaheuristic. The metaheuristic is tested on benchmark instances from the literature and on a real-life instance in which the trade-off between arrival time diversification and transportation costs is demonstrated.

The contribution of this study is fourfold. (1) A new solution approach is presented for the vehicle routing problem with arrival time diversification (VRPATD) that is capable of supporting decision making in various practical settings, e.g., for the transportation of valuable goods (Michallet et al., 2014; Yan et al., 2012) or for patrolling routes performed by security companies (Calvo & Cordone, 2003). (2) An efficient method is proposed to determine whether a route is time window feasible when multiple time windows are available per customer and when waiting time is not allowed. (3) Four different methods for penalizing violations of the multiple time windows and waiting time constraints are developed and tested. (4) The proposed iterated granular tabu search obtained new best known solutions to all benchmark instances generated in Michallet et al. (2014). Compared to the results in Michallet et al. (2014), total distance is reduced by 29% and computational time by 93% on average.

This paper is organized as follows. In Section 2, an overview of the literature related to routing diversification and multiple time windows is provided. In Section 3, the problem description and the mathematical model are presented. In Section 4, the handling of multiple time windows is described: the algorithm for determining route feasibility is proposed in Section 4.1, and different penalty functions are described in Section 4.4. The iterated granular tabu search is described in Section 5. The computational results are reported in Section 6, and the conclusions are presented in the last section.

2. Literature review

There is an extensive literature on constructing consistent routes to ensure that the same driver visits the same customers at approximately the same time on each day these customers need service (Coelho, Cordeau, & Laporte, 2012; Groër, Golden, & Wasil, 2009; Kovacs, Golden, Hartl, & Parragh, 2014). The converse problem of finding unpredictable routes is a relatively new topic. There are two streams in the literature, each focusing on a different aspect of route inconsistency: order diversification, and arrival time diversification.

The m-peripetic vehicle routing problem (m-PVRP) is the most restrictive way to model order diversification. The objective of this problem is to find a set of edge-disjoint routes of minimal cost over m periods such that each customer is visited exactly once per period. It is a generalization of the VRP and the m-peripatetic salesman problem (m-PSP). The m-PVRP was first studied in Ngueveu, Prins, and Calvo (2010), where lower and upper bounds for the problem were presented. A drawback of this model for practical application is that it is strict: if customers are served in the order $a - b$, then edge (a, b) is removed and the reverse order $b - a$ cannot be used anymore.

In Talarico, Sörensen, and Springael (2015a), this constraint was relaxed by formulating a similar problem called the k-dissimilar vehicle routing problem (kd-VRP). The goal is to find k different VRP solutions for which the similarity between a pair of solutions is below a certain threshold. Even though multiple use of an edge is allowed in the kd-VRP, it is restricted by a constraint on the similarity between the k solutions. In Talarico et al. (2015a), various similarity measures were presented to compare VRP solutions based on, among others, the number of shared edges and the cost of the shared edges. The objective function differs from the standard objective of minimizing distance, as it minimizes the worst-case cost across the k solutions.

In Akgün, Erkut, and Batta (2000), it was argued that edge diversification does not necessarily imply that routes are geographically spread. Geographically spread routes are used for the transport of hazardous materials, where the risk of an accident must be equally divided over the population. In Dell'Olmo, Gentili, and Scozzari (2005), spatially dissimilar paths were introduced by measuring the intersection of the areas surrounding paths. The buffer zone of a path is defined as the area obtained by moving a circle over the edge. In this method, a set of Pareto-optimal paths is obtained by solving a multi-criteria shortest path algorithm. Buffer zones for these paths are calculated using GIS, and then the subset of paths with maximal spatial dissimilarity is selected.

In Bozkaya, Salman, and Telciler (2017), a bi-objective function was used to minimize the transportation cost and the security risk of transporting cash. A composite risk measure was proposed that is the weighted sum of the predictability and the risk of a solution. Predictability is measured by the number of times an arc is used compared to the solutions on previous days. Risk is defined by a measure using the social-economic status of the neighborhoods through which the vehicles travel. In Talarico, Sörensen, and Springael (2015b) and Talarico, Sörensen, and Springael (2017), another risk measure was proposed that is proportional to the amount of cash carried and the distance covered by the vehicles.

The second approach to designing unpredictable routes consists in varying the arrival time at each customer over successive visits. In Calvo and Cordone (2003), the importance of differing arrival times was pointed out in the case of overnight security patrolling. It was argued that it is not the routing of the guards that matters but the time at which each location or customer is served. Therefore, the goal is to find a set of different solutions in terms of arrival times. It is assumed that all customers have the same service time window. First, the routing problem without arrival time constraints is solved and the corresponding assignments of the customers to the guards is fixed. Secondly, the customer's time window is split into four sub-time windows and the customers are divided into four groups based on their arrival times in the solution. Finally, for every guard, $4! = 24$ different solutions are created by changing the sub-window assigned to every customer set and solving the corresponding TSP with time windows. Thereby, 24 different solutions are created for each guard, one of which is randomly selected every night.

In Constantino, Mourão, and Pinto (2017), the problem of collecting the safes from parking meters was discussed. In this case, an arc represents a task; the goal is to minimize the total routing time and maintain a similarity index less than a certain value. To measure the similarity between tours on different days, each day is divided into a fixed number of periods containing a fixed number of tasks. The similarity index of two routes is defined as the percentage of tasks that are performed in the same period. A MILP model was formulated and a metaheuristic was proposed to solve this problem for H days simultaneously.

In the model proposed in Yan et al. (2012), the current arrival time at customer i should differ by at least β_p minutes from the p^{th} previous arrival time with $p \in \{1, \ldots, P\}$. This minimum arrival time difference decreases for arrival times further away in the past, i.e., $\beta_1 \geq \beta_2 \geq \beta_3 \geq \ldots$. For example, when the route for

Saturday is planned, the arrival time at customer i should differ by at least β_1, β_2, and β_3 minutes from the previous arrival times on Friday, Thursday, and Wednesday, respectively. Furthermore, it is allowed that δ_p% of the customers violate the arrival time constraints compared to the previous day p. The allowed percentage δ_p is smaller if the preceding day is closer. It should be noted that it may happen that the arrival time at the same customer is always violated. The problem is formulated as an integer multiple-commodity network flow problem. To solve instances of approximately 40 customers, the schedule day is divided into several time periods, thus creating multiple sub-problems that are solved individually by CPLEX. Owing to the limitations of standard solvers, the algorithm cannot handle larger-scale (practical) instances in a reasonable timespan.

To generate unpredictable routes in Michallet et al. (2014), a periodic VRPTW was presented in which customers must be visited daily and the arrival times must be spread across each customer's time window. The model does not allow waiting time, and subsequent visits to the same customer must differ by a given time constant ϵ. An iterated local search was proposed in which infeasible solutions are allowed during the search. Arrival times that violate the time window or the arrival time spread constraints are penalized at each customer by a piecewise linear function. Therefore, every departure time from the depot results in a different total penalty. Forward and backward costs are used to calculate the minimum penalty of a route and to efficiently evaluate local search operators. The forward cost and backward cost are the total penalty of serving a customer sequence σ if the last customer is served at time t and if the first customer is served at time t, respectively. Owing to the time-dependent penalty functions at each customer, these forward and backward costs are also time-dependent and should be calculated for every time point, which is a computationally intensive task. Furthermore, the periodic setting increases complexity, as a change in one route in a specific period can change penalties in other periods.

Only in Michallet et al. (2014) was the arrival time diversification problem solved for medium to large-sized instances of the VRP. Therefore, the problem instances and the solution method in that study will be used to benchmark the proposed solution approach. A solution method for the VRPATD is proposed that improves all results in Michallet et al. (2014). A rolling horizon approach of one day is proposed, rather than a periodic solution approach in combination with the time-dependent penalty function as in Michallet et al. (2014). In each period, arrival time slots from previous days are removed from the solution space, resulting in a vehicle routing problem with multiple time windows (VRPMTW). This reduces the problem complexity and the solution space compared to Michallet et al. (2014), making it easier to find promising solutions.

To the best of the authors' knowledge, there is no published study on routing problems with multiple time windows without waiting time. Recently, in Tricoire, Romauch, Doerner, and Hartl (2010), Belhaiza et al. (2014), and Hoogeboom et al. (2018), algorithms were presented to minimize the duration of a route by determining the optimal departure time from the depot for the VRPMTW, in which waiting time is allowed. In the present case, as waiting time is not allowed, the duration of a route is fixed because it consists only of the travel and service time. Furthermore, it should still be determined whether a departure time exists for which the route is feasible, i.e., in which every customer is served in one of its time windows without causing any waiting time. Existing algorithms for the VRPMTW that minimize duration are not efficient for the present problem. Therefore, a novel algorithm is proposed for determining the time window feasibility of a route for the VRPMTW without waiting time. The reader is referred to Hoogeboom et al. (2018) for a more extensive literature review on the VRPMTW.

3. Model formulation

In the vehicle routing problem with arrival time diversification (VRPATD), the current arrival time at customer i should differ by at least ϵ_i minutes from the P previous arrival times at customer i. Arrivals before the start of a time window are not allowed because waiting times are forbidden for security reasons. The CIT company plans the routes on a daily basis because in practice, replenishment orders are received on short notice. Therefore, the VRPATD is addressed as a rolling horizon VRPMTW of one day, in which the multiple time windows are constructed by deleting the P previous arrival times with bandwidth ϵ_i from the initial service time window of customer i.

The problem is defined on a complete directed graph $G = (V, A)$ with vertices $V = \{0, 1, \ldots, n, n+1\}$ and arc set $A = \{(i, j) \in V \times V : i \neq j\}$. The vertices $V' = \{1, \ldots, n\}$ correspond to the set of customers that should be served and nodes 0 and $n+1$ represent the depot. The set of customers that should be served may differ per day; therefore, $V'_\delta \subseteq V'$ represents the customers on day δ, and A_δ is the corresponding arc set. A non-negative travel time τ_{ij} is associated with each arc $(i, j) \in A$, and let K be the set of available homogeneous vehicles. The demand and service time of customer $i \in V'$ are given by q_i and s_i, respectively. The demand and service time of the depot are set to zero, i.e., $q_0 = q_{n+1} = s_0 = s_{n+1} = 0$. For all vertices $i \in V$, let $[e_i, l_i]$ be the initial service time window in which it is possible to start serving customer i. The capacity of a vehicle can be restricted for physical, security, or insurance reasons. Let Q denote the maximum value that an armored truck may carry. Furthermore, the duration of a route of an armored truck is limited by the working hours of the drivers, and is denoted by D.

Let α_{ip} be an input parameter representing the p^{th} previous arrival time at customer i, where $i \in V'$ and $p \in \{1, \ldots, P\}$. The current arrival time at customer i should differ by ϵ_i time units from the P previous arrival times at customer i. It should be noted that P and ϵ_i are input parameters, and the maximum length of ϵ_i depends on the length of the initial service time window and P as follows: $\epsilon_i = \lfloor \phi \times \epsilon_i^{\max} \rfloor$, where $\phi \in [0, 1]$ and $\epsilon_i^{\max} = \frac{l_i - e_i}{2P}$.

To illustrate the arrival time solution space at customer i, the example in Fig. 1 is considered. By removing intervals with radius ϵ_i around the previous arrival times, the solution space consists of multiple time windows in which the customer is still available for service, indicated by the thick green lines in Fig. 1. This implies that it is required to select one of the available time windows for each customer. Thereby, the problem can be formulated as a VRPMTW. Let $T_i = \{1, \ldots, |T_i|\}$ be the index set of the time windows of customer i and let $\{[e_i^1, l_i^1], \ldots, [e_i^{|T_i|}, l_i^{|T_i|}]\}$ be the non-overlapping time windows of customer i, with $e_i^1 \leq l_i^1 < e_i^2 \leq l_i^2 < \ldots \leq l_i^{|T_i|}$. These time windows result from deleting the previous arrival intervals, as shown in Fig. 1.

The VRPATD is solved in a rolling horizon framework of a single day in which the parameters are adjusted daily. When a solution is found for day δ with arrival time a_i at customer i, then the time windows of each customer $i \in V'_\delta$ should be adjusted for future visits. The update procedure is performed in two steps. First, the P^{th} previous arrival time taken into account at customer i in the solution for day δ will not be taken into account for future visits at customer i. As a result, the arrival time area around α_{iP} is feasible again and should be included in the time windows. Secondly, the interval with bandwidth ϵ_i around arrival time a_i at customer i on day δ is no longer feasible for future visits at customer i. This interval is therefore deleted from the arrival time solution space, i.e., from the time windows of customer i. The results of the two-step

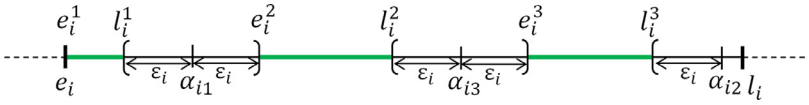

Fig. 1. An example of the arrival time solution space of customer i with $P = 3$.

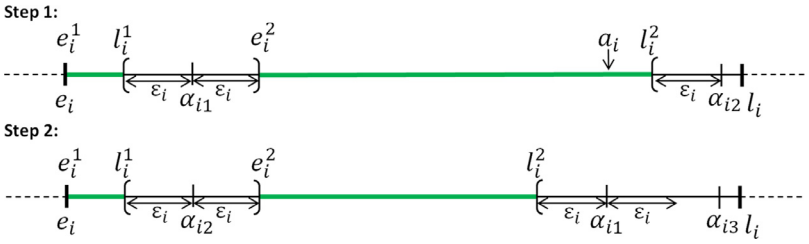

Fig. 2. An example of the update procedure of the arrival time solution space of customer i with $P = 3$.

update procedure for the example in Fig. 1 is shown in Fig. 2. In step 1, the area around the previous arrival time α_{i3} is included in the time windows. In step 2, the area around the new arrival time is blocked and the indices of the previous arrival times are updated. It should be noted that in step 2 the areas around α_{i1} and α_{i2} overlap, and therefore there are only two feasible time windows after the update.

To solve the VRPATD, an Iterated Granular Tabu Search (IGTS) is proposed. Before describing the IGTS, the implications of having multiple time windows will be explored in more detail. In the next section, a method will be presented to efficiently determine whether routes are feasible in terms of time windows, and different methods will be developed to penalize infeasible solutions. Both the proposed feasibility check and the penalization methods are embedded in the IGTS described in Section 5.

4. Handling multiple time windows

In traditional routing problems with multiple time windows, waiting time is allowed and the total duration is minimized. In such problems, the waiting time of a given route is minimized by determining the optimal departure time from the depot. In the present problem, waiting time is not allowed; thus, for all feasible departure times from the depot, the duration of a given route is the same, that is, it is equal to the sum of the service time and the travel time. However, it should still be verified whether a route is feasible because not all departure times from the depot are feasible, owing to the multiple time windows. In the present problem, a route is *feasible in terms of time windows* if there exists a departure time from the depot such that the vehicle arrives in one of the time windows at each customer in the route.

In this section, a method is first proposed to efficiently determine whether a route is feasible in terms of time windows. Subsequently, it is demonstrated that this algorithm can be used to determine whether neighborhood operations result in a time window feasible route. Then, four different methods are described for penalizing routes that are not feasible in terms of time windows.

4.1. Time window feasibility check

To determine if a route is time window feasible, we use the forward start intervals introduced in Hoogeboom et al. (2018) for the VRPMTW. In contrast to the problem in Hoogeboom et al. (2018), the VRPATD does not allow waiting time; therefore, the definition and generation of the forward start intervals should be adjusted.

Let σ be a route with m customers. For the sake of simplicity, let the customers be denoted by $\sigma' = \{1, \ldots, m\}$, and let 0 and $m+1$ represent the depot. The forward start intervals of customer i are the feasible service start times for customer i such that the preceding customers are also feasible, i.e., service starts in one of their time windows without any waiting time. Let F_i be the index set of the forward start intervals associated with customer $i \in \sigma'$. Let $[E_i^F(y), L_i^F(y)]$ be the forward start interval y of customer i. These intervals are non-overlapping and increasingly ordered, i.e., $E_i^F(1) \leq L_i^F(1) < E_i^F(2) \leq L_i^F(2) < \cdots < E_i^F(|F_i|) \leq L_i^F(|F_i|)$.

The forward start intervals are iteratively constructed from the first to the last customer in a route. Therefore, the forward start intervals of customer $i+1$ are obtained by comparing the forward start intervals of customer i with the time windows of customer $i+1$. The forward start interval $y \in F_i$ leads to a new forward start interval in the time window $t \in T_{i+1}$ at customer $i+1$ if $[E_i^F(y) + s_i + \tau_{i,i+1}, L_i^F(y) + s_i + \tau_{i,i+1}]$ and $[e_{i+1}^t, l_{i+1}^t]$ are overlapping. Hence, all feasible combinations between forward start intervals F_i and time windows T_{i+1} are given by $\{(y,t) \in F_i \times T_{i+1} | [E_i^F(y) + s_i + \tau_{i,i+1}, L_i^F(y) + s_i + \tau_{i,i+1}] \bigcap [e_{i+1}^t, l_{i+1}^t] \neq \emptyset\}$. The new forward start interval z at customer $i+1$ generated from the feasible combination (y, t) is calculated by the following forward recursion:

$$E_{i+1}^F(z) = \max\{E_i^F(y) + s_i + \tau_{i,i+1}, e_{i+1}^t\},$$
$$L_{i+1}^F(z) = \min\{L_i^F(y) + s_i + \tau_{i,i+1}, l_{i+1}^t\}. \quad (1)$$

The iterative process is initialized with the forward start interval of the depot, which is equal to the time window $[e_0, l_0]$. A route is feasible if the last customer in the route (customer m) has a non-empty set of forward start intervals. It should be noted that every forward start interval of a customer corresponds to a different selection of time windows. An illustrative example of generation of forward start intervals is shown in Fig. 3. In this example, the service times are set to zero and the lines between the customers indicate the travel time. The thick black lines represent the forward start intervals per customer.

As the time windows and forward start intervals are non-overlapping and increasingly ordered, several combinations between forward start intervals of customer i and time windows of customer $i+1$ can be excluded. This is shown in Proposition 1.

Proposition 1. *If the forward start interval $y \in F_i$ and the time window $t \in T_{i+1}$ are a feasible combination, then the forward start interval $y' \in F_i$, with $y' > y$, and the time window $t' \in T_{i+1}$, with $t' < t$, cannot be a feasible combination.*

Proof. Assuming that (y, t) is a feasible combination, we have $[E_i^F(y) + s_i + \tau_{i,i+1}, L_i^F(y) + s_i + \tau_{i,i+1}] \bigcap [e_{i+1}^t, l_{i+1}^t] \neq \emptyset$ by definition. Therefore, $e_{i+1}^t \leq L_i^F(y) + s_i + \tau_{i,i+1}$. Let $y' \in F_i$ and $t' \in T_{i+1}$, with $y' > y$ and $t' < t$, respectively. As the time windows and forward start intervals are non-overlapping and increasingly

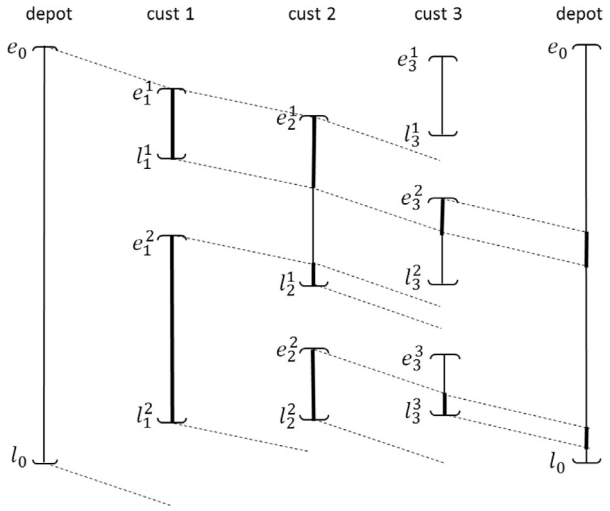

Fig. 3. Forward start intervals for the three customers in a route.

ordered, we have that $l^{t'}_{i+1} < e^t_{i+1} \leq L^F_i(y) + s_i + \tau_{i,i+1} < E^F_i(y') + s_i + \tau_{i,i+1}$. Hence, (y', t') is not a feasible combination. □

The Forward Algorithm (Algorithm 1) generates the forward start intervals in a depth-first manner. The algorithm stops as soon as a forward start interval is found for the last customer m, i.e., when a feasible schedule has been found for route σ. In Appendix A, it is shown that all feasible start times at customer i are included in a forward start interval. Therefore, the Forward Algorithm indicates the time window feasibility of the route $\sigma = \{0, 1, \ldots, m, m+1\}$.

Algorithm 1 Forward Algorithm.

```
1: Initialization: θ = {θ_1, ..., θ_m} = {1, ..., 1} & Feasible = false;
2: for t = 1 : |T_1| do
3:     FCC(1, t, e_0, l_0, θ, Feasible)
4:     if Feasible = true then
5:         return Feasible
6:     end if
7: end for
```

By the Feasible Combination Check (FCC, Algorithm 2), it is determined if the forward start interval $[E_{i-1}, L_{i-1}]$ of customer $i-1$ and the time window t of customer i are a feasible combination. If $E_{i-1} + s_{i-1} + \tau_{i-1,i} \leq l^t_i$ and $L_{i-1} + s_{i-1} + \tau_{i-1,i} \geq e^t_i$, then this is a feasible combination and the new forward start interval at customer i is calculated; then the process continues to the next customer. Only if $L_{i-1} + s_{i-1} + \tau_{i-1,i} > l^t_i$ can the forward start interval $[E_{i-1}, L_{i-1}]$ and the time window $t+1 \in T_i$ be a feasible combination. This is checked in lines 10–11 in Algorithm 2. If $E_{i-1} + s_{i-1} + \tau_{i-1,i} > l^t_i$, then a vehicle that serves customer $i-1$ during $[E_{i-1}, L_{i-1}]$ arrives after time window t at customer i; thus, this combination is not feasible, and the next time window $t+1$ at customer i should be checked.

It should be noted that in line 9, the process does not start from the first time window at customer $i+1$ but from the last checked time window θ_{i+1} at customer $i+1$, following Proposition 1.

By Proposition 1, the maximum number of feasible combinations between forward start intervals F_{i-1} and time windows T_i is equal to $|F_{i-1}| + |T_i| - 1$. Therefore, the worst-case complexity of the Forward Algorithm for the VRPATD is $O(\sum_{i=1}^{m}(1 + \sum_{j=1}^{i}(|T_j| - 1))$, as proven in Appendix A. This complexity is the same as for the algorithm proposed in Hoogeboom et al. (2018). However, the average computational time of the Forward Algorithm will be significantly less than that of the algorithm for the VRPMTW because waiting time is not allowed; therefore, fewer forward start intervals are generated and no dominance checks should be performed.

Algorithm 2 Feasible Combination Check $FCC(i, t, E_{i-1}, L_{i-1}, \theta, Feasible)$.

```
1: if Feasible = true then
2:     return Feasible
3: else if i ≤ m AND t ≤ |T_i| then
4:     if E_{i-1} + s_{i-1} + τ_{i-1,i} ≤ l_i^t then
5:         if L_{i-1} + s_{i-1} + τ_{i-1,i} ≥ e_i^t then  ▷ Arrival in time window t at customer i
6:             θ_i = t                                    ▷ Adjust last checked time window
7:             E_i = max{E_{i-1} + s_{i-1} + τ_{i-1,i}, e_i^t}
8:             L_i = min{L_{i-1} + s_{i-1} + τ_{i-1,i}, l_i^t}
9:             FCC(i + 1, θ_{i+1}, E_i, L_i, θ, Feasible)
10:            if L_{i-1} + s_{i-1} + τ_{i-1,i} > l_i^t then
11:                FCC(i, t + 1, E_{i-1}, L_{i-1}, θ, Feasible)
12:            end if
13:        end if
14:    else                                               ▷ Arrival after time window t
15:        FCC(i, t + 1, E_{i-1}, L_{i-1}, θ, Feasible)
16:    end if
17: else if i > m then
18:    Feasible = true
19: end if
```

To evaluate neighborhood operations during the local search quickly, all forward start intervals at all customers in a route are required. The breadth-first implementation that generates all forward start intervals, given in Appendix B, is faster than the depth-first implementation when all forward start intervals should be generated. The fast neighborhood evaluation will be demonstrated in Section 4.2, but the backward start intervals should first be defined.

4.1.1. Backward recursion

As shown in the previous section, the forward start intervals are sufficient to determine whether a route is time window feasible. To evaluate local search operations quickly, the backward start intervals are also required. The backward start intervals of customer i represent the service start times for customer i such that the succeeding customers are feasibly served, i.e., the service start times for customers $i+1, \ldots, m$ lie in one of their time windows without any waiting time. The construction of the backward start intervals is similar to the generation of the forward start intervals except that they are recursively generated from the last customer m to the first customer 1.

Let B_i be the index set of the backward start intervals of customer i, sorted in increasing order, and let $[E^B_i(y), L^B_i(y)]$ be the backward start interval y of customer i. If the backward start interval y of customer $i+1$ is compared with the time windows t of customer i, then this results in a new backward start interval of customer i if $[E^B_{i+1}(y) - \tau_{i,i+1} - s_i, L^B_{i+1}(y) - \tau_{i,i+1} - s_i]$ and $[e^t_i, l^t_i]$ are overlapping. Hence, the set of feasible combinations between backward start intervals B_{i+1} and time windows T_i is given by $\{(y,t) \in B_{i+1} \times T_i | [E^B_{i+1}(y) - \tau_{i,i+1} - s_i, L^B_{i+1}(y) - \tau_{i,i+1} - s_i] \cap [e^t_i, l^t_i] \neq \emptyset\}$. Let (y, t) be a feasible combination; then the new backward start interval z generated from (y, t) is given by

$$E^B_i(z) = \max\{E^B_{i+1}(y) - \tau_{i,i+1} - s_i, e^t_i\},$$
$$L^B_i(z) = \min\{L^B_{i+1}(y) - \tau_{i,i+1} - s_i, l^t_i\}. \quad (2)$$

The procedure is initialized with the backward start interval $[e_0, l_0]$ of the depot, and the generation of all backward start intervals

of all customers in a route can be performed as in the case of the forward start intervals.

4.2. Efficient feasibility check for local search moves

Once the forward and backward start intervals for all customers have been obtained, it can be quickly determined whether a local search operation results in a time window feasible solution. As most local search operations are based on insertion and deletion of customers, only these operations will be discussed.

4.2.1. Deletion

Assume that customer i is to be removed from route $\sigma = \{0, 1, \ldots, m, m+1\}$. As waiting time is not allowed, it may happen that route σ is feasible and the new route $\sigma' = \{0, 1, \ldots, i-1, i+1, \ldots, m+1\}$ is not feasible because removing customer i can introduce waiting time in the route. To determine if the new route σ' is time window feasible, the forward start intervals of customer $i-1$ are compared with the backward start intervals of customer $i+1$. Route σ' is time window feasible if there exist $f \in F_i$ and $b \in B_{i+1}$ such that

$$[E^F_{i-1}(f) + s_{i-1} + \tau_{i-1,i+1}, L^F_{i-1}(f) + s_{i-1} + \tau_{i-1,i+1}] \bigcap [E^B_{i+1}(b), L^B_{i+1}(b)] \neq \emptyset. \quad (3)$$

If customer i is deleted from route σ, then the forward and backward start intervals of the customers in the route should be recalculated. As a part of the route remains the same, only the forward start intervals of customers $i+1, \ldots, m$ and the backward start intervals of customers $1, \ldots, i-1$ should be recalculated.

4.2.2. Insertion

Insertion of a customer is performed similarly. Assume that customer α is inserted between customers i and $i+1$ in route σ. To determine whether the new route $\sigma' = \{0, 1, \ldots, i, \alpha, i+1, \ldots, m+1\}$ is time window feasible, the forward start intervals of customer α are calculated based on the forward start intervals of customer i. As in (3), the forward start intervals $f \in F_\alpha$ and the backward start intervals $b \in B_{i+1}$ are compared to determine whether this results in a feasible route.

The same approach can be used for the removal and insertion of a sequence of customers, as well as for more complex operators. For example, the 2-Opt* operation removes edge $(i, i+1)$ and edge $(j-1, j)$ of two different routes and replaces them by (i, j) and $(j-1, i+1)$, respectively. To determine whether this operation results in a time window feasible solution, only the forward start intervals of customer i should be compared with the backward start intervals of customer j; the same holds for $j-1$ and $i+1$.

4.3. Departure time from the depot

If a solution to the VRPATD has been obtained for day δ, then multiple feasible departure times from the depot are possible for every route in the solution. The arrival times of the customers in a route can be immediately derived by fixing the departure time from the depot. As described in Section 3, the arrival time at customer $i \in V'_\delta$ for day δ should be deleted from the arrival time solution space, along with the bandwidth ϵ_i. To ensure that the arrival time solution space for subsequent visits is as large as possible, the departure time from the depot is selected such that the total length of the intervals deleted at all customers in the route is minimized.

Let σ be a route in the solution for day δ and let σ' be the customers served in this route. Let $D_i(a_i)$ be the length of the deleted interval at customer $i \in \sigma'$ if the vehicle arrives at time a_i at customer i, i.e., $D_i(a_i) = \min(\epsilon_i, a_i - e^t_i) + \min(\epsilon_i, l^t_i - a_i)$ with $e^t_i \leq a_i \leq l^t_i$. In Fig. 4, two examples of the deleted interval of two possible arrival times a_i and a'_i at customer i are shown. As a'_i is scheduled close to the boundary of time window t, less than $2\epsilon_i$ is deleted from the solution space. The backward start intervals of the first customer in a route represent the service start times for the first customer so that all customers in the route are feasible. Therefore, the feasible departure times from the depot correspond to the backward start intervals of the first customer in a route, because if the service at the first customer starts at time χ, then the corresponding departure time is given by $\chi - \tau_{01}$.

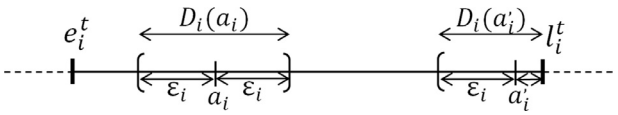

Fig. 4. The deleted intervals of two arrival times a_i and a'_i in time window t of customer i.

Assuming that route σ is feasible, there are $|B_1|$ backward start intervals at the first customer in the route, i.e., the start time at customer 1 should lie in one of these $|B_1|$ backward start intervals. Therefore, the optimal start time in $[E^B_1(y), L^B_1(y)]$, with $y \in B_1$, should be determined, i.e., the time that minimizes the total length of the deleted intervals of all customers in the route. If a vehicle serves the first customer at time χ, then the vehicle arrives at time $\chi + \sum_{j=1}^{i-1} s_j + \tau_{j,j+1}$ at customer i; thus, the total length of the deleted intervals at all customers in route σ is given by $D(\chi) = \sum_{i=1}^{m} D_i(\chi + \sum_{j=1}^{i-1} \tau_{j,j+1} + s_j)$. Lemma 1 shows that the optimal start time at customer 1 is obtained at one of the boundaries of the backward start intervals.

Lemma 1. *The start time χ of service for the first customer in a route that minimizes $D(\chi)$ is obtained at $E^B_1(1), L^B_1(1), \ldots, E^B_1(|B_1|)$, or $L^B_1(|B_1|)$.*

Proof. $\min(\epsilon_i, a_i - e^t_i)$ and $\min(\epsilon_i, l^t_i - a_i)$ are both concave piecewise linear functions on $a_i \in [E^B_i(y), L^B_i(y)]$ with $y \in B_i$. As the sum of two concave piecewise linear functions is again concave piecewise linear, $D_i(a_i)$ is a concave piecewise linear function on the interval $[E^B_i(y), L^B_i(y)]$. Therefore, $D(\chi)$ is also concave piecewise linear in $\chi \in [E^B_1(y), L^B_1(y)]$. Hence, the minimum of $D(\chi)$ in this interval is obtained at one of the endpoints, i.e., at $E^B_1(y)$ or $L^B_1(y)$. □

Therefore, for each route in the solution for day δ, the total deleted interval length $D(\chi)$ should be calculated for $\chi = E^B_1(1), L^B_1(1), \ldots, E^B_1(|B_1|), L^B_1(|B_1|)$, and the service start time for the first customer χ that minimizes $D(\chi)$ is selected.

4.4. Penalty functions for time window violation

In Section 4.1, it is demonstrated that the forward and backward start intervals can be used to determine whether a route is feasible in terms of time windows. It is often the case in metaheuristic searches that to allow the search to reach new areas of the solution space, some constraints may be violated at the expense of a penalty. In this section, different methods are presented for penalizing a solution in which the time windows are violated. A more efficient implementation of the penalty measure defined in Michallet et al. (2014) is presented, and three alternative penalty measures are proposed that are less time consuming than the penalty measure in that study. The four penalty methods can be divided in two categories corresponding to whether a vehicle waits at a customer or not.

Wait: This penalty method is based on the classical VRPTW with duration minimization from Savelsbergh (1992), in which waiting time is allowed but penalized. If a vehicle arrives at time a_i before a time window at customer i, then it waits until the start

of the time window. The penalty at this customer is equal to the waiting time, i.e., $p_i = e_i^t - a_i$ if $l_i^{t-1} < a_i < e_i^t$, where $l_i^0 = 0$. If a vehicle arrives after the last time window $[e_i^{|T_i|}, l_i^{|T_i|}]$ at customer i, then the delay at customer i is calculated by $p_i = a_i - l_i^{|T_i|}$. If the vehicle arrives within a time window, then the penalty is zero, i.e., $p_i = 0$. The total time window violation penalty of solution x is given by the sum of the waiting time and delay at every customer, namely, $t(x) = \sum_{i \in V} p_i$.

For every route in the solution, two options for the departure time from the depot are considered corresponding to two different penalty functions:

- Penalty 1: Departure from the depot so that the first customer is served as early as possible, i.e., the departure time of the depot is equal to $\max\{e_0, e_1^1 - \tau_{01}\}$.
- Penalty 2: Choose the departure time from the depot so that the penalty is minimized.

Do not wait: In this penalization method, all customers are immediately served upon arrival even if the arrival time lies outside the time windows of the customer. This penalty method is comparable to the time window violation penalties for the VRP with soft time windows (Figliozzi, 2010; Fu, Eglese, & Li, 2008). The total time window penalty is calculated by summing up the difference between the arrival time and the closest time window boundary (previous or next time window) at all customers that violate the time window constraint. Therefore, the total time window violation is measured by $t(x) = \sum_{i \in V} \min_{t \in T_i} |\min\{a_i - e_i^t, l_i^t - a_i, 0\}|$. Again, there are two options for the departure time from the depot:

- Penalty 3: Departure from the depot at $\max\{e_0, e_1^1 - \tau_{01}\}$ so that the first customer is serviced as early as possible.
- Penalty 4: Departure from the depot so that the penalty is minimized. This penalty is equal to the penalty measure used in Michallet et al. (2014).

As Penalties 1 and 3 involve fixed departure times from the depot, the calculations of these penalty functions are straightforward and easy to implement. Penalties 2 and 4 minimize the time window violations across all departure times from the depot. In the remainder of this section, a technique for efficiently obtaining this minimum penalty is indicated for both penalty methods. For explanatory reasons, Penalty 4 will be discussed first and then Penalty 2.

4.4.1. Do not wait: minimal penalty

Penalty function 4 corresponds to the penalty function proposed in Michallet et al. (2014). Let $p_i(a)$ be the penalty function of customer i representing the penalty when a vehicle arrives at times a at customer i, i.e., $p_i(a)$ is the difference between arrival time a and the closest time window at this customer. Two examples of penalty functions are given for Customer 1 and Customer 2 in Fig. 5. As these penalty functions are piecewise linear functions, they are completely determined by the breakpoints, represented by dots in Fig. 5. Let $R_i = \{r_i^1, \ldots, r_i^{|R_i|}\}$ represent the timestamps corresponding to the breakpoints of customer i, with $p_i(r_i^j)$ being the penalty at breakpoint r_i^j.

For every customer in a route, the forward penalty function F_i is generated. Let $F_i(a)$ be the total penalty at customers $\{1, \ldots, i\}$ when a vehicle arrives at time a at customer i. The forward penalty functions are recursively constructed by $F_i(a) = p_i(a) + F_{i-1}(a - s_{i-1} - t_{i-1,i})$. An example of the generation of the forward penalty function of Customer 2 is shown in Fig. 5.

In Michallet et al. (2014), it is stated that the forward penalties should be calculated for every time point. However, the forward penalty functions are piecewise linear functions and hence

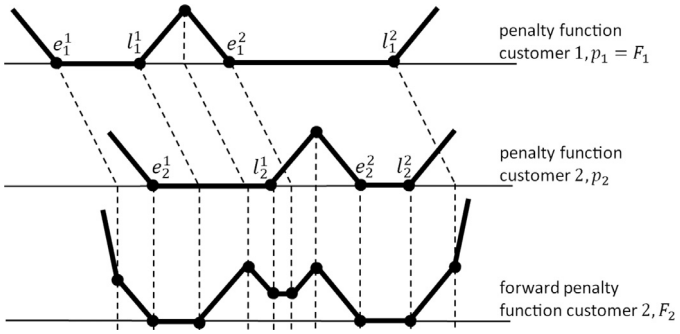

Fig. 5. An example of generating the forward penalty function of Customer 2.

completely determined by the breakpoints. Therefore, in the proposed approach, only the breakpoints of the forward penalty functions are considered, which may result in a significant reduction in computational time. Let $\bar{R}_i = \{f_i^1, \ldots, f_i^{|\bar{R}_i|}\}$ be the set of breakpoints corresponding to the forward penalty function F_i. Let $A_i = \{a_i^1, \ldots, a_i^{|\bar{R}_{i-1}|}\}$ be the set of arrival times of customer i corresponding to the breakpoints $\bar{R}_{i-1}$, with $a_i^k = r_{i-1}^k + s_{i-1} + \tau_{i-1,i}$ for $k = 1, \ldots, |\bar{R}_{i-1}|$. The arrival times A_i and the original breakpoints R_i of customer i form the breakpoint set $\bar{R}_i$ of the forward penalty function of customer i, i.e., $\bar{R}_i = A_i \bigcup R_i$. Only the forward penalties corresponding to these breakpoints are calculated by $F_i(f_i^j) = p_i(f_i^j) + F_{i-1}(f_i^j - s_{i-1} - t_{i-1,i})$ for all $f_i^j \in \bar{R}_i$. The forward penalties at other arrival times can be easily derived as follows: If $f_i^{j-1} < a < f_i^j$, then $F_i(a)$ is determined using linear interpolation of the breakpoints f_i^{j-1} and f_i^j. If the arrival time lies before the first breakpoint, i.e., if $a < f_i^1$, then all customers are served before their first time window. In this case, the penalty is equal to the penalty at the first breakpoint and the additional difference $f_i^1 - a$ that is encountered at all customers $1, \ldots, i$; thus, the penalty is equal to $F_i(a) = F_i(f_i^1) + i(f_i^1 - a)$. Similarly, if $a > f_i^{|\bar{R}_i|}$, then all customers are served after the last time window, and the penalty is equal to $F_i(a) = F_i(f_i^{|\bar{R}_i|}) + i(a - f_i^{|\bar{R}_i|})$.

The forward penalty function F_m of the last customer in a route represents the total penalty function of the entire route $\{1, \ldots, m\}$. Therefore, F_m can be used to calculate the minimum penalty of the route. As stated in Lemma 2, the minimum value of a forward penalty function is always obtained at a breakpoint. Hence, to calculate the minimum penalty of a route, only the breakpoints of the total penalty function F_m should be considered.

Lemma 2. *The minimum value of a forward penalty function is always obtained at a breakpoint.*

Proof. *(By contradiction.)* Let a be the arrival time at customer i with minimum penalty $F_i(a)$, and it is assumed that a is not a breakpoint of F_i. As the penalty functions of the customers increase outside the time windows, the forward penalty functions also increase before the first breakpoint and after the last breakpoint. Therefore, the arrival time a is located between two breakpoints, i.e., there exists j such that $f_i^j < a < f_i^{j+1}$. As the forward penalty functions are piecewise linear, the breakpoint f_i^j or f_i^{j+1} has a lower penalty than a or all three penalties are equal, i.e., $F_i(f_i^j) = F_i(a) = F_i(f_i^{j+1})$, which leads to a contradiction. Therefore, the minimum value of the forward penalty function F_i is always attained at a breakpoint. □

The backward penalty function $B_i(a)$ of customer i represents the total penalty of the customers $\{i, \ldots, m\}$ when a vehicle arrives at time a at customer i. The backward penalty functions are

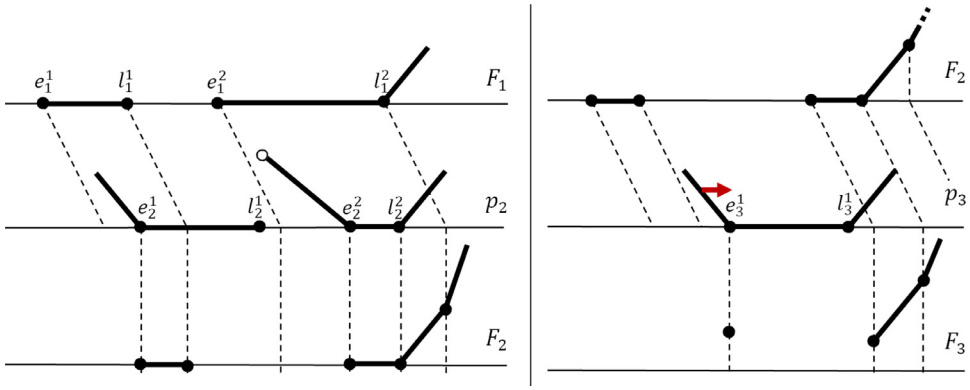

Fig. 6. Generating the forward penalty functions of Customer 2 (left) and Customer 3 (right).

generated as in the case of the forward penalty functions, but from the last to the first customer in a route.

The forward and backward penalty functions can be used to efficiently recalculate the lowest penalty of the route when a neighborhood operation is evaluated. For example, if customer i is deleted from route σ, then the total penalty function of the new route is obtained by summing the forward penalty of customer $i-1$ and the backward penalty of customer $i+1$. Hence, the total penalty function of the new route is given by $F(a) = F_{i-1}(a) + B_{i+1}(a + s_{i-1} + \tau_{i-1,i+1})$. As in Lemma 2, the minimum penalty of the new route is obtained at one of the breakpoints of the new total penalty function. Therefore, only the breakpoints of $F(a)$ should be considered.

4.4.2. Wait: minimal penalty

For Penalty function 2, the forward and backward penalty functions defined in the previous section are also used. However, as waiting time is now allowed, the generation of the functions should be adjusted. When a vehicle arrives between time windows, it should wait until the start of the next time window. Hence, the arrival and start times for serving a customer are not always the same as in Penalty 2. Let $p_i(a)$ be the penalty of customer i when a vehicle arrives at time a at customer i. Let $F_i(\chi)$ be the forward penalty of customers $\{1, \ldots, i\}$ when the service at customer i starts at time χ. The forward penalty functions are recursively generated from the first to the last customer in the route. Two examples of forward penalty function generation are given in Fig. 6. Owing to the waiting time, the (forward) penalty functions are not continuous; however, the functions are still piecewise linear, and therefore they are also completely determined by the breakpoints.

The service at customer i can start only in the time windows or after the last time window at the expense of a penalty. Let $dom(F_i)$ be the domain of $F_i(\chi)$ with $dom(F_i) \subset \{[e_i^1, l_i^1] \cup \ldots \cup [e_i^{|T_i|}, l_i^{|T_i|}] \cup [l_i^{|T_i|}, \infty[\}$. For $\chi - s_{i-1} - \tau_{i-1,i} \in dom(F_{i-1})$ the forward penalty function of customer i is generated as follows:

$$F_i(\chi) = \begin{cases} F_{i-1}(\chi - s_{i-1} - \tau_{i-1,i}) \\ F_{i-1}(\chi - s_{i-1} - \tau_{i-1,i}) + p_i(\chi) \end{cases}.$$

For other start times χ, $F_i(\chi)$ is not defined because then χ is not a feasible start time. As it is possible that a vehicle arrives between time windows and waits until the start of the next time window, the penalties corresponding to the start times of the time windows should also be calculated. Let $a_t = \max\{\chi \,|\, \chi \leq e_i^t$ and $\chi - s_{i-1} - \tau_{i-1,i} \in dom(F_{i-1})\}$ be the latest arrival time before the time window boundary e_i^t. If $l_i^{t-1} < a_t \leq e_i^t$, then this arrival time lies between the time windows $t-1$ and t; hence, the vehicle should wait, and thus the penalty at start time e_i^t is equal to $F_i(e_i^t) = F_{i-1}(a_t - s_{i-1} - \tau_{i-1,i}) + p_i(a_t)$. This penalty corresponds

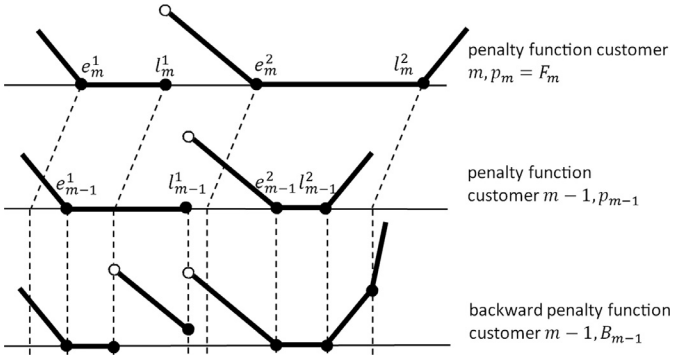

Fig. 7. Generating the backward penalty function of customer $m-1$.

to the waiting time and is represented by the horizontal arrow in the right part of Fig. 6. If a_t does not exist or $a_t \notin]l_i^{t-1}, e_i^t]$, then $e_i^t \notin dom(F_i)$.

To recalculate the minimum penalty quickly if neighborhood operations are applied, backward penalties are also required. Let the backward penalty function $B_i(a)$ of customer i represent the total penalty of the customer sequence $\{i, \ldots, m\}$ when a vehicle arrives at time a at customer i. The backward penalty function of all customers can be recursively calculated from the last to the first customer in a route by $B_i(a) = p_i(a) + B_{i+1}(a + s_i + \tau_{i,i+1})$ for $a \in [e_i^t, l_i^t]$, where $t \in T_i$ and $a > l_i^{|T_i|}$. Owing to the waiting time, the backward penalties for arrival times $l_i^{t-1} < a < e_i^t$ are given by $B_i(a) = B_i(e_i^t) + p_i(a) = B_i(e_i^t) + e_i^t - a$. An example of the generation of the backward penalty function of customer $m-1$ is given in Fig. 7. It should be noted that as the backward function represents the penalty for all arrival times instead of start times, there are no domain issues.

If, for example, customer i is to be deleted from route σ, then the total penalty function of the new route is given by $F(\chi) = F_{i-1}(\chi) + B_{i+1}(\chi + s_{i-1} + \tau_{i-1,i+1})$ for $\chi \in dom(F_{i-1})$. As the function $F(\chi)$ is again piecewise linear, the minimum penalty of the new route is obtained at a breakpoint of $F(\chi)$. Therefore, only these points should be considered to determine the minimum penalty of the new route.

5. Iterated granular tabu search heuristic

Tabu search is a well-known search strategy that explores the solution space and moves each iteration to the best solution in the neighborhood. To avoid being trapped in a local optimum, moving to poorer solutions is accepted if no improving solution is available in the neighborhood. To prevent cycling, a move that restores

the previously accepted move to a new solution will be prohibited for θ iterations. The tabu status can be overridden when the new solution is the best ever found, which is the so-called aspiration criterion (Gendreau, Hertz, & Laporte, 1994).

In this section, the components of the Iterated Granular Tabu Search (IGTS) are described. An overview of the IGTS heuristic is given in Algorithm 3. The IGTS is initialized with a feasible initial solution x^* constructed by applying the route minimization (RM) heuristic in Nagata and Braysy (2009), as described in Section 5.3. In each iteration, the solution $\bar{x}$ with minimum objective value $V(\bar{x})$ is selected from the neighborhood, and the local optimum $\bar{x}^*$ and the best found solution x^* are updated if necessary. If the local optimum $\bar{x}^*$ does not improve for η iterations, then the search is restarted by shaking the solution; details are described in Section 5.2. The IGTS stops if the maximum number of iterations I is reached or if the incumbent solution x^* is not improved for S iterations.

Algorithm 3 Iterated granular tabu search (IGTS).

```
Initialization: Create an initial feasible solution x̄ by the RM heuristic, x* = x̄* = x̄
for I iterations do
    Select x ∈ N(x̄) that minimizes V(x̄), set x̄ = x
    if V(x̄) < V(x̄*) then                ▷ Check if current solution improves local optimum
        x̄* = x̄
        if V(x̄) < V(x*) then            ▷ Check if current solution improves incumbent
            x* = x̄
        end if
    end if
    Update the tabu status
    Every μ iterations: Update the self-adjusting parameters β1, β2 and β3
    if x̄* not improving for η iterations then
        Shake solution x̄                ▷ Shaking phase is called
        x̄* = x̄                          ▷ Reset local optimum
    end if
    if x* not improving for S iterations then
        return x*
    end if
end for
```

As different operators are used in the shaking procedure, the heuristic bears a similarity to a VNS algorithm. Because no systematic change of neighborhoods is used in both the shaking procedure and the local search, the present method is referred to as an iterated granular tabu search.

5.1. Penalized objective function V(x)

Let x be a solution and let the objective value be equal to $c(x) = t(x) + Fm$, with $t(x)$ being the total travel time, F being the fixed cost of using a vehicle, and m being the number of vehicles used. Following (Gendreau et al., 1994) and Ho and Gendreau (2006), to obtain better feasible final solutions, the violation of the capacity, duration, and time window constraints is allowed during the search at the expense of a penalty. The overload $q(x)$ is calculated by $q(x) = \sum_{k=1}^{|K|} \max\{\sum_{(i,j)\in A} q_i x_{ij}^k - Q, 0\}$ and the overtime is given by $d(x) = \sum_{k=1}^{|K|} \max\{\sum_{(i,j)\in A}(s_i + \tau_{ij})x_{ij}^k - D, 0\}$. For the time window violation $t(x)$, four different penalty functions are described in Section 4.4. Therefore, the penalized objective function of the solution x is given by $V(x) = c(x) + \beta_1 q(x) + \beta_2 d(x) + \beta_3 t(x)$, where β_i are parameters that are self-adjusted every μ iterations. If in the last μ iterations the capacity constraint is violated in at least one route, then the parameter β_1 is multiplied by $\delta > 1$; otherwise, the capacity constraint holds in all μ solutions and β_1 is divided by δ. The same update rule is applied to parameters β_2 and β_3.

5.2. Granular neighborhood and tabu list

To accelerate the local search, a granular neighborhood will be used. This sparsification method was first introduced in Toth and Vigo (2003) and is used to improve calculation time by restricting the neighborhood to moves containing elements likely to be part of high-quality solutions. The restricted arc set in the present study consists of the arcs from each customer to the c% closest customers. All depot arcs are also included in the restricted arc set, as it was shown in Schneider, Schwahn, and Vigo (2017) that this improves solution quality. In every iteration, the best move of the composite neighborhood is selected. The composite neighborhood is obtained by using five well-known operators for the VRP with time windows:

Intra-route

- $\mathcal{N}_1$ - Relocate: Customer i is relocated to the best new position in the same route.
- $\mathcal{N}_2$ - Exchange: Two customers i and j from the same route exchange position.

Inter-route

- $\mathcal{N}_3$ - Relocate: Customer i is relocated to the best position in another route.
- $\mathcal{N}_4$ - Cross-exchange: Two customers i and j from two different routes exchange position.
- $\mathcal{N}_5$ - 2-Opt*: The tails of two routes are exchanged.

As described in Toth and Vigo (2003), in the tabu search, only the moves with a generator arc in the restricted arc set are performed. Let (i, i^+) be the generator arc, where i^+ is the new succeeding node following customer i after one of the operators above is applied. In Fig. 8, the generator arc of the cross-exchange and 2-Opt* moves are shown in bold. After the generator arc for a given operator is chosen, the other arcs involved in a move follow immediately. This implies that arcs that do not belong to the restricted arc set can be inserted by the operators as well. During the search, a different tabu status is used for the intra- and inter-route operators. For the inter-route moves, the attribute set of a solution as introduced in Cordeau, Laporte, Mercier et al. (2001) is used. The attribute set of a solution x is given by $B(x) = \{(i, k) : \text{customer } i \text{ is visited by vehicle } k\}$. Therefore, the attribute set $B(x)$ defines the structure of the solution x and can be used to control the tabu status. As in Belhaiza et al. (2014), the set L_θ contains the attribute sets of the θ most recently explored solutions. A solution x generated by an inter-route operator is declared tabu if $B(x) \in L_\theta$. An intra-route operation does not change the allocation of customers to vehicles. Therefore, the attribute set remains the same, and a different tabu status will be used: if an intra-route move is performed on customer i in vehicle k, then intra-route moves involving customer i in route k are declared tabu for θ iterations. If a tabu move improves the best known solution, then the aspiration criterion is satisfied and the tabu move is performed.

A shaking procedure is applied when the local optimum does not improve for η iterations. For the shaking procedure, one operator is randomly selected out of four operators: the first two operators are the intra-exchange and 2-Opt*. The third operator performs relocation of a sequence of (at most) four customers to the best position in another route. The last is the cross-sequence operator in which two sequences of at most four customers of two different routes exchange positions. For the inter-route operators, the

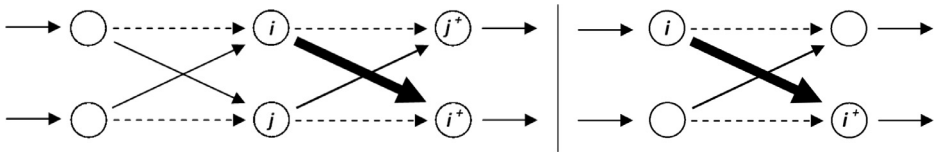

Fig. 8. Example of Cross-exchange (left) and 2-Opt* (right): the generator arc is shown in bold and the dashed arcs represent the removed arcs.

best move of every combination of non-empty routes is executed. The intra-route operator is applied to every non-empty route, and a minimum of S_{itr} moves should be performed.

If the incumbent solution x^* has not improved after ρ shakes, then the solution is restarted from the incumbent solution and the current tabu lists and penalties are retained. This will diversify the search, as it is highly likely that the local search will take a different path (Schneider et al., 2017).

5.3. Initial solution

To construct an initial solution, each customer is first served by a separate vehicle. To minimize the number of vehicles, the RM heuristic in Nagata and Braysy (2009) is used. In each iteration, the RM heuristic attempts to reduce the number of routes by selecting a route from the current solution for removal. The customers of this removed route are put in the ejection pool introduced in Lim and Zhang (2007). First, attempts are made to insert a customer v of the ejection pool in the remaining routes without violating capacity, duration, and time window constraints. If there is no feasible insertion position for customer v, then the customer is inserted in the position that minimizes the penalized objective function $V(x)$ defined in Section 5.1. Secondly, it is attempted to restore the infeasible solution by local search moves that minimize the penalty. If this second approach fails, then up to three customers are removed from the existing routes to create a feasible insertion position for customer v. The ejected customers are added to the ejection pool. The ejected customers are selected using the concept of guided local search (Voudouris & Tsang, 2003), which estimates the difficulty of re-inserting the ejected customers. Finally, after the ejection of customers, the search is diversified by M tabu search moves from a randomly selected neighborhood $(\mathcal{N}_1, \mathcal{N}_2, \mathcal{N}_3, \mathcal{N}_4, \mathcal{N}_5)$. These steps are repeated until EP=$\emptyset$ is reached, and then the next route is selected. The RM heuristic stops if the number of vehicles is equal to the lower bound $\lceil \sum_{i=1}^{N} \frac{q_i}{Q} \rceil$ or if the maximum execution time T^{max} is reached.

6. Computational results

In this section, the IGTS is first used to evaluate the effect on solution quality and running time of the penalty functions proposed in Section 4.4. Subsequently, the best performing penalty is used to compare the performance of the IGTS with the results in Michallet et al. (2014). Then, a real-life instance is used to illustrate the trade-off between arrival time diversification and transportation costs.

The algorithms were implemented in C++ and executed on a Windows 7 computer with a 2.3 GHz Intel core i5-6200U and 8 GB RAM. As in Michallet et al. (2014), the (Solomon, 1987) VRPTW instances were used to test the IGTS. The instances were adjusted by adding an arrival time spread value ϵ_i per customer, a number P of previous arrival times taken into account, and the number of days in the problem instance (period). It should be noted that every day, the same set of customers was considered. A simple preprocessing of the time windows was performed to adjust the lower and upper bound in case of conflict with the time window of the depot, i.e., $e_i = \max\{e_i, e_0 + \tau_{0i}\}$ and $l_i = \min\{l_i, l_0 - \tau_{i0} - s_i\}$ for all $i \in V'$. To tune the IGTS parameters, the same penalty function as in Michallet et al. (2014) was used (Penalty 4), but with the more efficient implementation. The tuning strategy described in Ropke and Pisinger (2006) was applied to set the parameters of the IGTS. The initial parameter values were chosen based on preliminary results during the development of the heuristic and on parameter settings used in the literature. During the tuning, one parameter changed value, whereas the other parameters were fixed. For each parameter, five runs were conducted on a randomly selected subset of the instance set (with two instances from each of the sets C1, C2, R1, R2, RC1, and RC2), and the parameter value with the best average results was selected. This process was repeated for all parameters and resulted in the following parameter setting: The vehicle cost F was set to 400 and the solutions were kept tabu for $\theta = 50$ iterations. The shaking procedure was called after $\eta = 150$ non-improvement iterations in which at least $S_{itr} = M = 10$ moves were performed. The search restarted from the incumbent after $\rho = 4$ shakes were executed. The granular neighborhood of every customer consisted of the $c = 50\%$ closest customers and the depot arcs. The penalty parameters were adjusted every $\mu = 20$ iterations with factor $\delta = 1.2$, and the parameters were initialized at $\beta_1 = \beta_2 = \beta_3 = 100$. The maximum number of iterations was set to $I = 10,000$ and the stopping criterion to $S = 4,000$, resulting in a good trade-off between computational time and solution quality. The same holds for the running time of the RM heuristic, which was set to $T^{max} = 30$ seconds.

6.1. Influence of penalty functions

In the first set of computational experiments, the penalty functions described in Section 4.4 are evaluated in terms of their performance. In the first two penalization methods, the vehicle waits if it arrives at a customer between time windows. If the vehicle arrives after the last time window at a customer, then the delay is calculated. The penalty is given by the sum of the waiting time and delay. In Penalties 3 and 4, all customers are immediately served upon arrival even if the arrival time lies outside the time windows of the customer. In this case, the difference between the arrival time and the closest time window boundary (previous or next time window) is penalized. In Penalties 1 and 3, the vehicle departs from the depot so that the first customer in the route is served as early as possible. The optimal departure time that minimizes the penalty is used in penalization methods 2 and 4.

The four penalty methods were used in the IGTS and were tested on the (Solomon, 1987) VRPTW instances for a three-day period. The value $P = 2$ was considered; thus, only on day 3 should two previous arrival times be taken into account. Furthermore, $\epsilon_i = \lfloor 0.5 \times \epsilon_i^{max} \rfloor$ with $\epsilon_i^{\max} = \frac{l_i - e_i}{2P}$. For every penalty type, the IGTS was performed three times, and the average results are shown in Table 1. Per instance set, the average number of vehicles used and the average total distance are shown in columns “nVeh” and “distance”, respectively.

In the column “No penalty”, the results for the IGTS are given in which infeasible solutions are not allowed during the search. In this case, the IGTS stops if there is no feasible solution available in the composite neighborhood. In the other columns, the results are

Table 1
Average number of vehicles and average total distance of the IGTS using different penalty functions.

	No penalty		Wait				Not Wait			
			Penalty 1		Penalty 2		Penalty 3		Penalty 4	
Instance	nVeh	distance	nVeh	distance	nVeh	distance	nVeh	distance	nVeh	distance
C1	30.5	3413.4	30.4	3005.6	30.4	2937.9	30.3	2964.2	30.4	2906.0
C2	9.9	2815.0	9.9	2365.9	9.9	2320.5	9.9	2365.9	10.1	2367.4
R1	39.2	4424.4	39.1	4323.2	39.0	4333.5	39.1	4313.5	39.1	4314.4
R2	9.2	3910.0	9.0	3586.1	9.2	3566.3	9.1	3618.9	9.2	3561.1
RC1	37.4	4833.9	37.5	4740.3	37.5	4738.9	37.3	4757.5	37.5	4771.5
RC2	10.3	4917.1	10.3	4496.6	10.2	4522.9	10.2	4534.6	10.5	4498.5
Average	23.35	4059.8	23.29	3771.4	23.27	3755.9	23.24	3777.0	23.39	3753.5

Table 2
For the different penalty functions the average, best, and standard deviation of the objective value are presented. The average computational time is reported in the last column.

		Best	Average	std	Time(sec)
No infeasible solution		13244.1	13400.3	143.1	69.7
Wait	P1: Fixed	12948.8	13085.7	122.1	134.8
	P2: Optimal	12963.8	13063.1	85.7	164.1
Not Wait	P3: Fixed	12960.2	13072.2	92.1	132.7
	P4: Optimal	12964.4	13108.3	128.4	492.8

given for the IGTS using one of the four penalty functions during the search. It should be noted that the final solution in the table is always feasible. The results demonstrate that allowing infeasible solutions during the local search reduces the average distance by 7–8%, depending on the penalty type. For both the "Wait" and "Do Not Wait" methods, the penalty that selects the optimal departure time has a lower average distance than Penalty 1 and Penalty 3, respectively. However, there is a trade-off between this more extensive calculation of the penalty and computational time, as shown in Table 2.

In Table 2, the best and average objective values are given for all penalty functions in columns "best" and "average", respectively. The objective value is the combination of the total distance and the vehicle cost. The standard deviation is given in column "std", and the average computational time in the last column. Penalty 4 is the most computationally intensive: the computational time is at least three times as high compared to that of the other penalty functions. As the RM heuristic has a time limit of T^{max} seconds, the number of vehicles is also slightly higher for Penalty 4. Penalty 2 exhibits the best trade-off between solution quality, computational time, and solution robustness and was therefore used in the remaining experiments.

6.2. Comparison to Michallet et al. (2014)

To examine the performance of the model formulation as a rolling horizon VRPMTW and the proposed IGTS solution approach, the results by the method were compared with those in Michallet et al. (2014), where the same arrival time diversification problem was investigated, but it was solved in a periodic setting. To test their model, the instances in Solomon (1987) with a period of three days were used. To align the present setting to that in Michallet et al. (2014), a period of three days was considered, and therefore P was set to 2. The same ϵ value was used as in Michallet et al. (2014), i.e., $\epsilon = \lfloor 0.5 \times \frac{\min_{i \in V'}\{l_i - e_i\}}{2} \rfloor$, which was used for all customers. Furthermore, in Michallet et al. (2014), the fleet size was increased to 30 vehicles to find a feasible solution for all instances. As in Michallet et al. (2014) only distance was minimized, in the present study, the vehicle cost was set to $F = 0$ to make the results comparable. The tests in Michallet et al. (2014) were performed on a 32-bit 2.8 GHz Intel core i5 computer with 4GB RAM compared to the 2.3 GHz Intel core i5 computer with 8GB RAM that was used in the present study.

In Table 3 the results by the proposed IGTS and the results in Michallet et al. (2014) are given per instance set. As in Michallet et al. (2014), the IGTS was run three times, and the best and the average distance are given in columns "best" and "average", respectively. The average computational time in seconds over the three runs is given in column "time", and the gaps are presented in the last two columns (reported by Michallet et al. (2014)-IGTS/Michallet et al., 2014). The results demonstrate that the IGTS significantly improved the results in Michallet et al. (2014), with an average distance reduction of 29% and an average computational time decrease of 93%. All instances were improved, with a maximum improvement of 62% and a minimum improvement of 9%. The new best known solutions per instance are given in Appendix C.

There could be various reasons for this improvement in both computational time and solution quality. First, in Michallet et al. (2014) Penalty function 4 was used, which was shown to be the most time consuming penalty in Table 2. Moreover, the penalty method was implemented less efficiently than in the present study, that is, by calculating the forward and backward cost for every time point, instead of using only the breakpoints. Secondly, in Michallet et al. (2014), a periodic setting was used in which a change in one period could have an impact on the feasibility and penalties in other periods; therefore, additional checks should be performed. Thirdly, because in Michallet et al. (2014) a periodic setting was used, the solution space was larger, and thus better solutions could potentially be obtained. However, such solutions were not found, as shown in Table 3, because the periodic setting seems to make the problem too complex for the solution approach used.

6.3. Real-world instance

Finally, the trade-off between arrival time diversification (inconsistency level) and transportation costs was examined in a real-life setting. A real world instance was provided by a cash management company in the Netherlands. It consisted of the requests of 400 ATMs and bank offices in March 2016. The customers were located in a region in the east part of the Netherlands, and there was a total of 26 replenishment days, because no transport is performed on Sundays. Some customers should be replenished every day, whereas others only once during this period. A real road distance matrix was used, and an average travel speed of 40 km/h was applied. The maximum duration of a trip was 8 hours. The time windows of the bank offices were set to 9:00–17:00 and the time windows of the ATMs and the depots were 7:00–19:00. The goal was to minimize the number of vehicles and the distance, with a vehicle cost of 400 EUR per truck.

Table 3
The results of the proposed IGTS and of Michallet et al. (2014) per instance set.

	IGTS			Michallet et al. (2014)			Gap	
Instance	Best	Average	Time	Best	Average	Time	Best (%)	Time (%)
C1	2613.2	2700.7	47.2	4185.7	4209.8	1037.1	−38	−95
C2	1924.5	2101.8	154.8	3351.9	3366.0	2590.0	−43	−94
R1	4287.7	4350.3	122.1	5561.6	5580.5	1302.3	−23	−91
R2	3444.7	3519.0	309.8	4738.4	4755.6	5269.6	−27	−94
RC1	4728.1	4812.6	120.0	6361.9	6383.0	980.9	−26	−88
RC2	4599.4	4711.6	249.5	6383.9	6406.3	3914.9	−28	−94
Average	3622.8	3718.3	169.5	5094.9	5114.4	2550.2	−29	−93

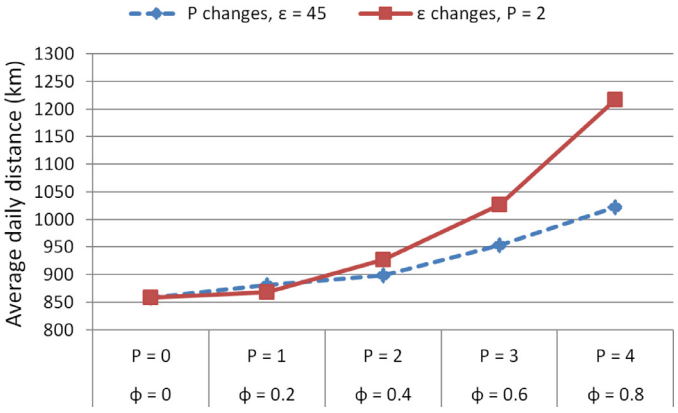

Fig. 9. Average daily distance for different inconsistency settings.

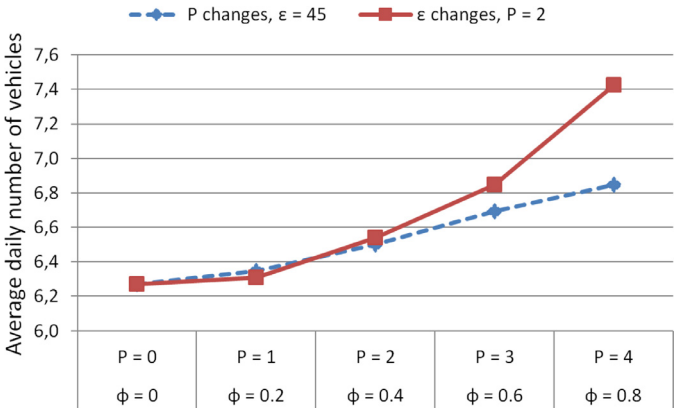

Fig. 10. Average daily number of vehicles used for different inconsistency settings.

The average daily distance was calculated for different values of the arrival time spread ϵ_i and for different numbers P of previous arrival times taken into account. The values of ϵ_i change by fluctuating ϕ, with $\epsilon_i = \lfloor \phi \times \epsilon_i^{max} \rfloor$. In Fig. 9, the relation between the average daily distance and inconsistency level is shown, where the inconsistency level increases if P or ϕ increases. For the solid line, the number of previous arrival times taken into account is fixed at $P = 2$ and the value of ϕ varies. The graph shows that the relation between inconsistency level and total distance is not linear: compared to the unconstrained problem ($\phi = 0$), the distance increases by 1.2% if $\phi = 0.2$ and by more than 8% if $\phi = 0.4$. If $\phi = 0.8$, then the total distance increases by more than 40% compared to the unconstrained problem.

This effect is weaker but also noticeable if ϵ_i is fixed at 45 minutes for all customers and the number of previous arrival times taken into account varies, as shown by the dashed line. A similar relationship between the average daily number of vehicles used and the inconsistency level is shown in Fig. 10. Therefore, this case illustrates that introducing arrival time diversification with low epsilon and P values can be achieved with low impact on the transportation cost. Clearly, setting high values for the inconsistency parameters ϕ and P can have a large impact on the cost.

6.3.1. Trade-off on Solomon Instances

The same experiment can be conducted on the Solomon instances. In Tables 4 and 5, the average increase in daily travel time and daily number of used vehicles is presented, compared to the unconstrained problem, for different values of the inconsistency parameters ϕ and P, respectively. In Table 4, P was set to 2, and in Table 5, ϵ_i was set to $\lfloor 0.5 \times \frac{l_i - e_i}{2 \times 4} \rfloor$ to keep the value of ϵ_i fixed for all values of P. On average, the increase in travel time and number of vehicles is higher for the set 2 instances. These are instances with a long planning horizon, and therefore the impact of the inconsistency parameters on the transportation cost is larger on instances with longer routes. Furthermore, when the length of the original service time windows increases, the impact of the inconsistency parameters increases as well. This is because the size of the arrival time solution space that is blocked increases when the service time window is larger.

The increase in daily travel time and number of vehicles is approximately linear in the number of past arrival times P. When the value of ϕ is increased, there is a jump in the cost from $\phi = 0.4$ to $\phi = 0.6$. This confirms the finding that for low values of ϕ and P, the impact on the cost is low, but it increases rapidly for higher values of the inconsistency parameters.

7. Conclusions and future research

Sufficiently unpredictable routes are required for the transport of valuable goods and for security patrolling. Accordingly, there is a growing interest in designing unpredictable routes (Calvo & Cordone, 2003; Michallet et al., 2014; Talarico et al., 2015a; Yan et al., 2012). The current study presents a novel method to diversify the arrival times of customers and to minimize transportation costs. The arrival times are diversified by removing the previous arrival times with surrounding bandwidth ϵ from the solution space. This results in a set of multiple time windows available to serve each customer. Therefore, the problem is formulated as a vehicle routing problem with multiple time windows that is solved in a rolling horizon of one day. As no periodic setting or computationally intensive penalty function is used, the proposed approach is easier, more efficient, and more powerful than existing methods such as that in Michallet et al. (2014). The IGTS proposed to solve the routing problem obtained new best-known solutions for all benchmark instances in Michallet et al. (2014). The average improvement in total distance was 29% and the computational time decreased by 93%.

Furthermore, because waiting times are not allowed for the problem under consideration, a method was proposed to efficiently determine if a route is time window feasible. To allow time window violations during the local search, four different penalty func-

Table 4
Increase in average daily travel time and average number of vehicles used for different values of ϕ compared to the unconstraint problem $\phi = 0$.

	Daily travel time				Daily vehicles used			
Instance	ϕ=0.2 (%)	ϕ=0.4 (%)	ϕ=0.6 (%)	ϕ=0.8 (%)	ϕ=0.2 (%)	ϕ=0.4 (%)	ϕ=0.6 (%)	ϕ=0.8 (%)
C1	7	7	28	39	0	0	4	8
C2	14	15	43	61	7	7	13	22
R1	2	3	11	16	3	3	10	15
R2	3	3	16	24	6	7	11	20
RC1	3	3	12	16	3	3	12	17
RC2	5	5	18	32	3	2	8	13
Average	5	6	20	30	4	4	10	16

Table 5
Increase in average daily travel time and average number of vehicles used for different values of P compared to the unconstraint problem $P = 0$.

	Daily travel time				Daily vehicles used			
Instance	P = 1 (%)	P = 2 (%)	P = 3 (%)	P = 4 (%)	P = 1(%)	P = 2 (%)	P = 3 (%)	P = 4 (%)
C1	4	9	15	22	0	0	1	1
C2	9	16	23	29	2	6	8	12
R1	1	4	6	10	2	3	6	8
R2	0	5	8	14	8	11	14	16
RC1	0	3	5	7	3	4	6	8
RC2	3	7	13	17	4	4	4	6
Total	3	7	11	16	3	5	7	9

tions were proposed and compared in terms of solution quality and computational time. Finally, computational experiments on a real-life instance quantified the trade-off between arrival time diversification and transportation costs.

Following (Yan et al., 2012), the proposed model can be extended by making ϵ_i period-dependent, so that ϵ_i decreases for arrival times further away in the past. Furthermore, the number of previous arrival times taken into account can also be allowed to deviate per customer. For example, if a customer is rarely visited, then fewer previous arrival times should be taken into account. These extensions influence only the number of intervals and the length of the multiple time windows; they do not change the proposed problem or solution method.

In theory, the periodic setting of Michallet et al. (2014) should obtain a better solution, as the solution space is larger. Therefore, it would be interesting to extend the proposed approach to a periodic setting. As route unpredictability is a relatively new area of research, there are several other topics to be considered in the future, e.g., congestion could be incorporated, or order and arrival time diversification could be combined in one approach. Comparing arrival time and order diversification approaches in terms of both inconsistency level and transportation costs could also be an interesting topic for future research.

Funding

This work was supported by The Dutch Science Foundation [grant number 407-13-050].

Acknowledgements

The authors would like to thank the unnamed cash management company for providing the real-life data set. Furthermore, the authors are grateful to the reviewers for their constructive comments that helped improve the presentation of the manuscript.

Appendix A. Proof of optimality and complexity for the Forward Algorithm

Herein, it will be first shown that every feasible service start time for customer i in route $\sigma = \{0, 1, \ldots, m, m+1\}$ lies in a forward start interval of customer i. Then, the complexity of the Forward Algorithm will be examined.

Lemma 3. *For all customers of route $\sigma = \{0, \ldots, i, \ldots, m+1\}$, all feasible start times at customer $i \in \sigma'$ are included in a forward start interval of i.*

Algorithm 4 Forward Algorithm (FA).

Input: $F_1 = T_1$ and $F_i = \emptyset\ \forall i \in \{2, \ldots, m\}$
for $i \in \{2, \ldots, m\}$ **do**
  $\theta = 1$
  for $y \in F_{i-1}$ **do**
    for $t \in \{\theta, \ldots, |T_i|\}$ **do**
      if $E^F_{i-1}(y) + s_{i-1} + \tau_{i-1,i} \le l^t_i$ **then**
        if $L^F_{i-1}(y) + s_{i-1} + \tau_{i-1,i} \ge e^t_i$ **then** ▷ Create a new forward start interval
          $z = |F_i| + 1$
          $E^F_i(z) = \max\{E^F_{i-1}(y) + s_{i-1} + \tau_{i-1,i}, e^t_i\}$
          $L^F_i(z) = \min\{L^F_{i-1}(y) + s_{i-1} + \tau_{i-1,i}, l^t_i\}$
        end if
        if $L^F_{i-1}(y) + s_{i-1} + \tau_{i-1,i} \le l^t_i$ **then** ▷ Go to next forward start interval
          $\theta = t$ ▷ Set the last visited time window
          break
        end if
      end if
    end for t
  end for y
end for i
Output: $F_i\ \forall i \in \{1, \ldots, m\}$

Table C1
New best-known solution per instance.

	Michallet et al. (2014)			IGTS - distance				IGTS - distance + vehicle cost				
Inst	best	average	time	nVeh	best	Obj av	time	nVeh	distance	Obj av	time	gap (%)
c101	3029.2	3038.2	401	31	2584.5	2584.5	63.1	31	2584.5	14984.5	62.8	−15
c102	4862.2	4868.9	1247	31	2993.6	3565.8	51.3	30	3458.3	15752.3	51.0	−38
c103	5485.1	5521.24	1559	30	2520.4	2681.1	42.3	30	2567.2	14614.1	45.4	−54
c104	5191.8	5224.88	2996	30	2534.2	2543.0	55.4	30	2522.8	14538.0	54.5	−51
c105	3157.1	3180.64	406	30	2528.2	2528.2	24.8	30	2528.2	14528.2	24.6	−20
c106	3527.5	3544.69	533	31	2610.5	2610.5	67.7	31	2610.5	15010.5	67.6	−26
c107	3833.6	3833.89	468	30	2528.2	2528.2	31.7	30	2528.2	14528.2	31.7	−34
c108	3902.3	3977.57	765	31	2568.5	2573.7	50.6	30	2582.5	14840.4	52.7	−34
c109	4682.3	4698.42	959	31	2651.1	2691.2	52.8	31	2692.8	15131.0	52.4	−43
c201	1987.5	1987.46	1716	11	1807.4	1807.4	147.4	10	1856.7	5856.7	146.2	−9
c202	3558.5	3563.59	1980	10	1807.1	1807.1	136.3	9	1810.3	5410.3	133.6	−49
c203	5362.9	5395.05	3295	10	2002.0	2116.9	195.9	9	1989.0	5664.8	194.5	−63
c204	3892.6	3907.27	4353	9	2105.8	2117.6	247.7	9	2105.8	5717.6	245.7	−46
c205	3137.3	3139.77	1696	12	1958.4	2229.9	134.5	11	1996.0	6764.6	131.8	−38
c206	2892.7	2894.48	2504	10	1861.7	1985.7	130.7	10	1861.7	5985.7	133.3	−36
c207	2648.6	2662.55	3185	10	1831.2	1837.8	148.8	9	1839.4	5538.6	147.1	−31
c208	3335.2	3378.01	1991	10	2022.2	2912.0	131.9	10	1974.8	6822.5	141.0	−39
r101	8427.5	8430.75	531	68	7296.0	7409.9	102.1	68	7482.5	35010.0	100.7	−13
r102	7695.0	7721.54	910	61	5852.9	5896.3	105.8	59	5830.0	29638.0	106.4	−24
r103	6299.2	6319.67	1489	44	4386.4	4432.6	119.3	44	4337.8	21972.5	116.6	−30
r104	4505.8	4564.06	1893	31	3257.5	3287.1	143.5	30	3298.2	15530.2	138.0	−28
r105	6101.4	6119.17	763	48	4960.6	5094.6	109.1	48	5099.5	24655.7	106.9	−19
r105	5597.3	5612.56	971	43	4276.0	4379.6	116.7	41	4382.1	21204.4	118.4	−24
r107	5154.0	5180.19	1639	34	3745.1	3831.5	125.8	33	3794.3	17273.5	136.6	−27
r108	4056.5	4060.53	1676	28	3183.5	3201.3	140.9	28	3189.0	14595.6	139.0	−22
r109	5123.2	5132.92	1009	37	4112.5	4131.1	120.5	37	4035.5	18893.1	118.9	−20
r110	4785.1	4804.51	1860	32	3584.1	3632.5	137.2	32	3546.0	16438.0	126.8	−25
r111	4742.1	4758.13	1819	32	3490.8	3564.4	130.5	31	3637.3	16209.0	128.5	−26
r112	4252.3	4261.69	1067	32	3306.8	3342.4	125.6	31	3423.9	15949.1	121.3	−22
r201	6127.6	6134.42	3255	14	4837.0	4943.3	225.9	12	4888.8	9847.6	213.0	−21
r202	5735.9	5760.71	5074	11	4130.9	4195.3	270.1	10	4196.0	8381.9	282.3	−28
r203	4909.7	4943.03	6028	9	3379.9	3431.3	330.3	9	3368.1	6979.4	355.0	−31
r204	3270.1	3302.31	6904	6	2677.2	2721.6	536.8	6	2677.2	5255.0	501.0	−18
r205	5673.3	5694.75	3486	11	4311.5	4476.3	242.8	10	4432.5	8576.1	238.4	−24
r206	4837.4	4846.25	4094	9	3760.0	3891.0	266.8	9	3898.7	7541.0	283.4	−22
r207	4625.5	4644.08	5041	8	3166.5	3239.0	328.3	8	3131.5	6403.9	325.4	−32
r208	3314.5	3325.32	7570	7	2595.4	2602.3	340.1	7	2595.4	5401.0	332.9	−22
r209	4664.8	4668.46	5484	9	2997.0	3057.5	279.1	9	3049.5	6663.3	273.7	−36
r210	5000.3	5015.57	4885	9	3289.5	3352.9	308.6	9	3397.7	7003.3	308.8	−34
r211	3963.6	3977.18	6145	9	2746.8	2798.5	287.8	8	2805.8	6138.1	316.0	−31
rc101	8427.5	8430.75	531	53	6353.1	6475.2	105.9	53	6423.8	27773.3	104.1	−25
rc102	7695.0	7721.54	910	45	5391.7	5519.8	111.6	45	5372.1	23811.1	110.4	−30
rc103	5943.3	5965.26	716	35	4325.4	4408.3	122.1	34	4556.0	18489.4	124.2	−27
rc104	5062.0	5074.7	1500	30	3662.4	3685.0	138.3	31	3613.3	16075.2	141.1	−28
rc105	7100.3	7121.3	862	40	5311.5	5412.9	109.1	40	5316.7	21502.7	108.9	−25
rc106	6555.3	6589.34	696	41	4986.1	5070.4	114.7	41	4994.0	21650.4	114.1	−24
rc107	5623.6	5638.54	1163	34	4194.4	4242.8	131.1	34	4137.9	17806.3	130.1	−25
rc108	4488.2	4522.39	1469	32	3600.1	3686.3	138.3	31	3706.8	16130.2	135.4	−20
rc201	8286.8	8314.42	1605	19	6725.6	6858.1	173.4	17	7210.7	14077.4	175.1	−19
rc202	7380.4	7409.24	3058	12	5588.1	5735.6	210.2	12	5573.9	10529.1	213.8	−24
rc203	6255.5	6269.33	4576	10	4505.4	4581.9	268.5	10	4245.0	8546.8	265.3	−28
rc204	4483.7	4510.67	5257	9	3033.2	3102.3	298.1	9	3049.1	6694.1	325.5	−32
rc205	7787.7	7812.84	2466	12	5282.8	5338.0	219.2	12	5463.3	10425.2	211.3	−32
rc206	6678.2	6682.86	2401	11	4832.7	5193.1	219.0	10	4991.0	9383.1	198.7	−28
rc207	5527.5	5542.67	5337	9	3745.9	3760.8	288.6	9	3680.7	7353.5	296.3	−32
rc208	4671.4	4708.56	6619	9	3081.9	3123.1	279.0	9	3070.8	6725.7	290.8	−34

Proof. The proof is by induction. For the first customer $i = 1$ in a route, the forward start intervals are equal to the time windows (after the preprocessing described in Section 6), and thus by definition, the lemma holds for $i = 1$. Assuming that the lemma holds for $i \leq j < m$, it suffices to show that the lemma holds for $i = j + 1$.

We assume that ζ_{j+1} is a feasible start time at customer $j + 1$. As ζ_{j+1} is feasible, ζ_{j+1} lies in one of the time windows of customer $j + 1$, i.e., there exists $t \in T_{j+1}$ such that $\zeta_{j+1} \in [e^t_{j+1}, l^t_{j+1}]$. Let $\zeta_j = \zeta_{j+1} - \tau_{j,j+1} - s_j$ be the service start time for customer j. By the induction hypothesis, ζ_j is feasible only if it lies in a forward start interval, i.e., there exists $y \in F_j$ such that $\zeta_j \in [E^F_j(y), L^F_j(y)]$. This implies that (y, t) is a feasible combination generating a new forward start interval at customer $j + 1$ that includes ζ_{j+1}. Hence, the lemma is true for all customers in σ. □

Lemma 4. *The computational complexity of the Forward Algorithm is* $O(\sum_{i=1}^{m}(1 + \sum_{j=1}^{i}(|T_j| - 1)))$

Proof. We will first show that the maximum number of times that the FFC algorithm is called to check a feasible combination between customers i and $i + 1$ is equal to $|F_i| + |T_{i+1}| - 1$. We define a bipartite graph $G = (F_i, T_{i+1})$, where F_i is the index set of the forward start intervals at customer i and T_{i+1} is the index set of the time windows of customer $i + 1$. There exists an edge between $q \in F_i$ and $t \in T_{i+1}$ if the combination (q, t) is checked by the FCC algorithm. Therefore, the number of arcs represents the maximum

number of calls to the FCC algorithm and the maximum number of forward start intervals of customer $i+1$. By putting the index set of the forward start intervals F_i and the time windows T_{i+1} both in increasing order, the bipartite graph G can be drawn without crossings following Proposition 1. As G is a bipartite graph without crossings, it does not contain cycles. Hence, G consists of a union of disjoint trees, which has at most $|F_i| + |T_{i+1}| - 1$ edges. As the maximum number of forward start intervals of customer 1 is $|F_1| = |T_1|$, by induction we obtain $|F_i| \leq 1 + \sum_{j=1}^{i}(|T_j| - 1)$ for all $i \in \sigma'$. The worst-case complexity occurs when the maximum number of forward start intervals is generated for customers $1, \ldots, m-1$, but no feasible combination exists between the forward start intervals of customer $m-1$ and the time windows of customer m. Therefore, this worst-case complexity is equal to $\sum_{i=1}^{m}(1 + \sum_{j=1}^{i}(|T_j| - 1))$. □

Appendix B. Breadth-first implementation

To calculate all forward start intervals of all customers $\{2, \ldots, m\}$. First all forward start intervals of customer 2 are generated, then of customer 3 and so on.

Appendix C. Solutions per instance

Per instance the new best solutions and average solutions over three runs are given in Table C1. First, the results reported by Michallet et al. (2014) are presented, with in column "best", "average", and "time" the best distance, average distance and average calculation time, respectively. The results of the proposed IGTS with the objective function solely consisting of the distance ($F = 0$) and with the objective consisting of distance and vehicle cost ($F = 400$) are reported. In column "nVeh" and "best" the number of vehicles used and the total distance are presented for the best solutions over three runs. The average objective value and average calculation time are represented in columns "Obj av" and "time". The last column represents the distance gap between the best solution of Michallet et al. (2014) and the IGTS with objective minimizing distance.

References

Akgün, V., Erkut, E., & Batta, R. (2000). On finding dissimilar paths. *European Journal of Operational Research, 121*(2), 232–246.

Belhaiza, S., Hansen, P., & Laporte, G. (2014). A hybrid variable neighborhood tabu search heuristic for the vehicle routing problem with multiple time windows. *Computers & Operations Research, 52*, 269–281. doi:10.1016/j.cor.2013.08.010.

Bozkaya, B., Salman, F. S., & Telciler, K. (2017). An adaptive and diversified vehicle routing approach to reducing the security risk of cash-in-transit operations. *Networks, 69*(3), 256–269.

Calvo, R. W., & Cordone, R. (2003). A heuristic approach to the overnight security service problem. *Computers & Operations Research, 30*(9), 1269–1287.

Coelho, L. C., Cordeau, J.-F., & Laporte, G. (2012). Consistency in multi-vehicle inventory-routing. *Transportation Research Part C: Emerging Technologies, 24*, 270–287.

Constantino, M., Mourão, M., & Pinto, L. S. (2017). Dissimilar arc routing problems. *Networks, 70*, 233–245.

Cordeau, J.-F., Laporte, G., Mercier, A., et al. (2001). A unified tabu search heuristic for vehicle routing problems with time windows. *Journal of the Operational research society, 52*(8), 928–936.

Dell'Olmo, P., Gentili, M., & Scozzari, A. (2005). On finding dissimilar pareto-optimal paths. *European Journal of Operational Research, 162*(1), 70–82.

Figliozzi, M. A. (2010). An iterative route construction and improvement algorithm for the vehicle routing problem with soft time windows. *Transportation Research Part C: Emerging Technologies, 18*(5), 668–679.

Fu, Z., Eglese, R., & Li, L. Y. (2008). A unified tabu search algorithm for vehicle routing problems with soft time windows. *Journal of the Operational Research Society, 59*(5), 663–673.

Gendreau, M., Hertz, A., & Laporte, G. (1994). A tabu search heuristic for the vehicle routing problem. *Management science, 40*(10), 1276–1290.

Groër, C., Golden, B., & Wasil, E. (2009). The consistent vehicle routing problem. *Manufacturing & service operations management, 11*(4), 630–643.

Ho, S. C., & Gendreau, M. (2006). Path relinking for the vehicle routing problem. *J Heuristics, 12*(1–2), 55–72. doi:10.1007/s10732-006-4192-1.

Hoogeboom, M., Dullaert, W., Lai, D., & Vigo, D. (2018). Efficient neighborhood evaluations for the vehicle routing problem with multiple time windows. *Technical Report OR-18-2 DEI University of Bologna, Italy*.

Kovacs, A. A., Golden, B. L., Hartl, R. F., & Parragh, S. N. (2014). The generalized consistent vehicle routing problem. *Transportation Science, 49*(4), 796–816.

Lim, A., & Zhang, X. W. (2007). A two-stage heuristic with ejection pools and generalized ejection chains for the vehicle routing problem with time windows. *Informs Journal on Computing, 19*(3), 443–457. doi:10.1287/ijoc.1060.0186.

Michallet, J., Prins, C., Amodeo, L., Yalaoui, F., & Vitry, G. (2014). Multi-start iterated local search for the periodic vehicle routing problem with time windows and time spread constraints on services. *Comput Oper Res, 41*, 196–207. doi:10.1016/j.cor.2013.07.025.

Nagata, Y., & Braysy, O. (2009). A powerful route minimization heuristic for the vehicle routing problem with time windows. *Operations Research Letters, 37*(5), 333–338. doi:10.1016/j.orl.2009.04.006.

Ngueveu, S. U., Prins, C., & Calvo, R. W. (2010). Lower and upper bounds for the m-peripatetic vehicle routing problem. *4OR, 8*(4), 387–406.

Ropke, S., & Pisinger, D. (2006). An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. *Transportation science, 40*(4), 455–472.

Savelsbergh, M. W. (1992). The vehicle routing problem with time windows: Minimizing route duration. *ORSA journal on computing, 4*(2), 146–154.

Schneider, M., Schwahn, F., & Vigo, D. (2017). Designing granular solution methods for routing problems with time windows. *European Journal of Operational Research*.

Solomon, M. M. (1987). Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations research, 35*(2), 254–265.

Talarico, L., Sörensen, K., & Springael, J. (2015a). The k-dissimilar vehicle routing problem. *European Journal of Operational Research, 244*(1), 129–140.

Talarico, L., Sörensen, K., & Springael, J. (2015b). Metaheuristics for the risk-constrained cash-in-transit vehicle routing problem. *European Journal of Operational Research, 244*(2), 457–470.

Talarico, L., Sörensen, K., & Springael, J. (2017). A biobjective decision model to increase security and reduce travel costs in the cash-in-transit sector. *International Transactions in Operational Research, 24*(1–2), 59–76.

Toth, P., & Vigo, D. (2003). The granular tabu search and its application to the vehicle-routing problem. *Informs Journal on computing, 15*(4), 333–346.

Tricoire, F., Romauch, M., Doerner, K. F., & Hartl, R. F. (2010). Heuristics for the multi-period orienteering problem with multiple time windows. *Comput Oper Res, 37*(2), 351–367. doi:10.1016/j.cor.2009.05.012.

Voudouris, C., & Tsang, E. P. (2003). Chapter 7 Guided local search. *Handbook of metaheuristics*. Springer.

Yan, S., Wang, S.-S., & Wu, M.-W. (2012). A model with a solution algorithm for the cash transportation vehicle routing and scheduling problem. *Computers & Industrial Engineering, 63*(2), 464–473.